Desarrollo y edición de juegos serios con Unity y E-Adventure

David Martín-Maldonado Jiménez

Javier Sandoval Ferrandis

Trabajo de fin de grado del Grado en Ingeniería Informática
Facultad de informática
Universidad Complutense de Madrid

Septiembre 2017

Director:
Manuel Freire Morán

Colaboradores:
Víctor Manuel Pérez Colado e Iván Pérez Colado
(uAdventure)

Índice

RESUMEN

Desde hace unas décadas y cada vez más, el entretenimiento de los jóvenes estudiantes suelen ser videojuegos. Ya sean en PC, en videoconsolas, en consolas portátiles o en Smartphone, estos juegos siempre han tenido la peculiaridad de engancharnos durante horas alejándonos del mundo real. Aprovechando esto surgieron los juegos serios (*Serious Games*), utilizando este entretenimiento para aplicar algo de formación y sacarle más beneficios que el de meramente divertir.

En este contexto entra el grupo de investigación e-UCM, que con el fin de fomentar estos juegos serios, desarrolló la iniciativa *eAdventure*, lanzada en 2007, para facilitar la creación de *Serious Games* y que cualquier docente pueda crear un juego sin tener conocimientos de programación.

En 2017, surge *uAdventure* a modo de actualización para aprovechar la versatilidad y potencia de *Unity*. Además de actualizar, este proyecto incluye mejoras gráficas y nuevas funcionalidades como la geolocalización.

Por otro lado, es interesante conjuntar este género de videojuegos con los minijuegos. Aquí es donde entra este Trabajo de Fin de Grado. Trataremos de ampliar la herramienta *uAdventure* aportando un módulo que permita al docente introducir pequeños minijuegos dentro de su aventura gráfica educativa, como rompecabezas que resolver para avanzar en el videojuego. Este proyecto tendrá valor, no sólo en sí mismo, sino también porque documenta el camino para futuros desarrolladores que quieran añadir nuevos tipos de minijuegos o, incluso, nuevos módulos en *uAdventure*.

Palabras claves: Unity 3D, juegos serios, e-learning, edición de juegos, minijuegos, plugin

ABSTRACT

For decades and ever more, young students' main source of entertainment has been videogames. Whether on PC, portable console or smartphone, these games have often the peculiarity of hooking us in for hours and taking us away from the real world. Taking advantage of this, Serious Games emerged using this form of entertainment for training purposes and to get more benefit than just having fun.

In this context, the e-UCM research group, in order to promote these Serious Games, developed the *eAdventure Initiative*, launched in 2007, to facilitate the creation of Serious Games and that any teacher can create a game without any previous knowledge of programming.

In 2017, *uAdventure* emerges as a reboot to take advantage of the versatility and power of *Unity*. In addition, this project includes graphical improvements and new features such as geolocation.

On the other hand, it is interesting to combine this genre of videogames with minigames. And here is where this Final Degree Project comes in. We will try to expand the *uAdventure* tool by providing a new module that allows the teacher to introduce small minigames into their graphic adventure, such as puzzles to solve in order to progress in the game. Not only will this project have value in itself, but also because it documents the guidelines for future developers who want to add new types of minigames or even new modules in *uAdventure*.

Keywords: Unity 3D, serious games, e-learning, games edition, minigames, plugin.

CAPÍTULO I: MOTIVACIÓN Y OBJETIVOS

Motivación

La labor del grupo *e-UCM* es acercar más la educación a las nuevas tecnologías y sobre todo acercarla a los estudiantes, procurando que aprender resulte más un pasatiempo y menos una aburrida obligación. A lo largo de la carrera, en la facultad siempre hemos tenido que hacer prácticas que consistían en implementar juegos, ya sea un CodeBreaker, un CandyCrush, un 4 en raya, etc… Este sistema siempre nos ha parecido muy atractivo, hasta el punto de elegir hacer juegos en las prácticas con temática libre. Al oír hablar de *eAdventure*, nos pareció una forma muy original de aumentar el interés del alumnado por la asignatura que fuera. Nos atrajo especialmente la idea de poder contribuir a este proyecto que influirá en la docencia a cualquier nivel, facilitando la creación y edición de juegos educativos, o *Serious Games*, que ayuden en la enseñanza.

Siempre será más entretenido aprender historia por medio de una aventura gráfica que recitando y resumiendo largos textos. Estas aventuras gráficas son juegos por escenas relativamente estáticas en los que el jugador interactúa con objetos y personajes, sin tener presión de tiempo (por ejemplo, el 'Monkey Island', ver Fig. 1).

Fig. 1: Escena del juego Monkey Island.

Dentro de toda aventura gráfica, es inevitable encontrarse con momentos en los que hay que resolver algún rompecabezas para avanzar en la historia, así como descifrar códigos para abrir puertas o buscar pistas que nos ayuden más adelante en la aventura. En el desarrollo de *Serious Games*, podemos incluir estos momentos de variedad como minijuegos.

Un minijuego es un juego muy sencillo y con sus propias normas que forma parte de un videojuego, como mero entretenimiento o como parte necesaria para realizar ciertas acciones especiales [1]. Estos minijuegos dentro de cualquier aventura gráfica pueden

servir de distracción de la trama principal, o pueden formar la trama en sí, como una suma de minijuegos (por ejemplo, "*Mario Party*").

En el caso de los *Serious Games*, si el jugador se centra únicamente en la trama principal tiene el riesgo de perder el interés y el juego puede caer en la monotonía. Los minijuegos en este proyecto servirán de estímulo y de refresco para la mente del jugador, pudiendo añadir carga educativa por sí mismos.

Objetivos

Con el fin de contribuir en el proyecto *uAdventure* y ayudar a hacer más accesible y divertida la educación para los estudiantes de todos los niveles, este Trabajo de Fin de Grado desarrollará una extensión de la herramienta *uAdventure* para incluir minijuegos dentro de las aventuras gráficas.

El objetivo principal de este proceso (Ob1), es poder generar minijuegos desde el editor de *uAdventure* y a su vez, interpretarlos. Se pretende crear una interfaz propia de editor de minijuegos y extender el intérprete de *uAdventure* para poder ejecutar estos minijuegos. Además de este objetivo, existen otros tanto propios del proyecto como personales.

Uno de estos objetivos, es poder desarrollar este *plugin* de la forma mejor estructurada posible (Ob2). Al tratarse de software libre, la implementación debe ser clara y ampliable, y esto es un objetivo necesario de nuestro trabajo. Queríamos que nuestra aportación al proyecto *uAdventure* fuera perfectamente entendible y fácilmente mejorable, de forma que fuera sencillo añadir nuevos tipos de minijuegos al editor y que el intérprete los trate correctamente.

Desde el principio, este trabajo nos pareció un gran reto que nos ayudaría a mejorar como programadores y formarnos en el entorno de *Unity* más allá del nivel usuario. Trabajar y formar parte de un proyecto real como *uAdventure* nos dará la oportunidad de mejorar nuestra percepción y aplicación de las técnicas de la Ingeniería del Software, como seguir un guión preestablecido con plazos de tiempo para el cumplimiento de distintos hitos (Ob3). Un proyecto necesita una fase de investigación del tema a tratar, una fase de documentación de lo necesario para el desarrollo y una planificación del trabajo que se hará, cómo se hará y con qué plazos se hará. Y esto antes de comenzar a programar.

En resumen, podemos decir que este Trabajo de Fin de Grado tiene 3 objetivos principales:

- Ob.1: Crear un *plugin* para editar y ejecutar minijuegos con *uAdventure*.
- Ob.2: Realizar una estructura legible y una buena documentación que faciliten futuras ampliaciones de *uAdventure*.
- Ob.3: Mejorar nuestro entendimiento de la Ingeniería del Software y nuestra percepción de las buenas prácticas de programación.

CAPÍTULO II: CONTEXTO DEL PROYECTO

En este capítulo, vamos a exponer la situación de este Trabajo de Fin de Grado, o lo que es lo mismo, explicar los antecedentes y entornos que atañen al desarrollo de este *plugin*.

Serious Games

El concepto de "juegos serios" se estableció mucho antes de la era informática, concretamente en 1970, cuando definió el concepto en su libro '*Serious Games*' [2] de la siguiente manera: "Una definición más convencional es aquella en la que un juego es un contexto con reglas entre adversarios que intentan conseguir objetivos. Nos interesan los juegos serios porque tienen un propósito educativo explícito y cuidadosamente planeado, y porque no están pensados para ser jugados únicamente por diversión.".

El considerado primer juego serio fue el *Army Battlezone* [3] que diseñó la compañía Atari en 1980 para adaptar su videojuego *Battlezone* para el entrenamiento militar.

Hasta finales de los años 90, el desarrollo de juegos educativos sufría continuos fracasos debido a su baja rentabilidad. En esta situación, algunos estudiosos comenzaron a examinar la utilidad de los juegos para otros fines además del entretenimiento. En los primeros años de la década del 2000, surgieron diversas organizaciones orientadas a impulsar el desarrollo de estos juegos serios con distintas temáticas formativas. En 2002, se creó en Washington DC la 'Serious Games Initiative' (Iniciativa Serious Games) con el objetivo de fomentar este desarrollo en temas políticos y de gestión. Poco después, aparecerían otras iniciativas para el panorama social ('Games for Change') y sanitario ('Games for Health').

Aunque pueda parecer algo subjetivo hacer una clasificación de los distintos tipos de juegos serios, ya se hizo una separación [4] en 5 categorías distintas:

- '*Advergaming*' o juegos publicitarios ("Everquest II", tiene una opción para hacer pedidos a *Pizza Hut*);
- '*Edutainment*' o juegos educativos ("La gran aventura de las palabras" (Anaya Interactiva – enero 1997), ver Fig. 2);
- '*Edumarket*' que es mezcla de varios aspectos combinados ("Food Force");
- '*Political games*' o juegos de denuncia ("Darfur is Dying");
- '*Training*' y simuladores ("Sim city", "The Sims" o "Flight Simulator").

Fig. 2: Escena de ‘La gran aventura de las palabras’

Grupo e-UCM

Fig. 3: logotipo del grupo e-UCM

e-UCM learning group es un equipo de investigación de la Facultad de Informática de la UCM [5], orientado al desarrollo de juegos serios, es decir, juegos educativos que facilitan tanto la enseñanza como el aprendizaje. Este grupo de investigación ha llevado a cabo numerosos proyectos *e-learning* en colaboración con otras instituciones como la Comunidad de Madrid, la UCM o la Universidad de Harvard.

Entre otros, el grupo participa en el proyecto RAGE relacionado con analíticas de aprendizaje en juegos serios. Estas analíticas de aprendizaje (‘learning analytics’) pueden definirse como la colección y análisis de los datos generados durante el proceso de aprendizaje con el fin de mejorar tanto el aprendizaje como la enseñanza [6]. El grupo, en los últimos tiempos, trata de aplicar estas analíticas al aprendizaje con juegos serios [7].

Proyecto eAdventure

La plataforma *eAdventure* [8] es uno de los proyectos estrella del grupo *e-UCM*, y fomenta la inclusión de *Serious Games* dentro del proceso educativo. Dado que éstos son a menudo más amenos y atractivos para los estudiantes, *eAdventure* quiere ofrecer la oportunidad a los distintos niveles de docencia, de poder mejorar la experiencia de sus estudiantes creando videojuegos educativos relacionados con su disciplina. En el proyecto intervienen expertos en Ingeniería del Software e informáticos de múltiples campos, para crear una plataforma que permita a los docentes, por medio del motor de juegos *eAdventure* y el editor gráfico *eAdventure*, crear estas aventuras gráficas educativas sin necesidad de ningún tipo de conocimiento en programación.

Fig. 4: logotipo de la plataforma eAdventure

Esta iniciativa se basa en los juegos del tipo "point-and-click" (también conocidos como aventuras gráficas) para el desarrollo de los SG (*Serious Games*). Estas aventuras gráficas se caracterizan por la calma y el estudio del entorno en busca de la solución. Esto ayuda mucho a equilibrar la balanza entre aprendizaje y entretenimiento, puesto que si hay mucho entretenimiento no se aprende y si hay demasiada información, el estudiante no se entretiene y puede no sentirse lo suficientemente motivado.

El proyecto *eAdventure* comenzó en 2007 y la primera versión se publicó a principios de 2008, y desde entonces ha sido revisada y mejorada por el grupo *e-UCM*. Construída sobre *Java*, esta herramienta trabaja como editor y motor de juegos. Hoy en día, al existir otros motores de videojuegos mucho más potentes como *Unity* o *Unreal Engine*, es posible hacer uso de sus herramientas como motor de juegos (ver Proyecto uAdventure). Además, la mayoría de navegadores, por falta de actualizaciones de seguridad, han dejado obsoletos los *Applets* de *Java*, como *Chrome* anunció en abril de 2015 [9] y Firefox en octubre de ese año [10]. Esto provocó que *Oracle* marcase como deprecado el *plugin* de *Java* para navegador [11].

Proyecto uAdventure

Frente a estos y otros factores, el grupo *e-UCM* decidió comenzar un nuevo proyecto a modo de actualización de *eAdventure*, basado en el motor de juegos *Unity*, llamado *uAdventure*. *Unity 3D* [12] fue elegido, entre otras cosas, por su soporte multiplataforma, ya que permite desarrollar aplicaciones para escritorio (Mac, Linux o Windows), para consolas (Wii, PlayStation o Xbox), para web y para móvil (Android y iOS). Algunos de los recientes juegos más populares han sido desarrollados con *Unity*, como "Pokemon GO" de Nintendo o "HearthStone" de Blizzard.

Fig. 5: Logotipo de uAdventure

Este proyecto tenía varios objetivos claros como la compatibilidad con otros SG elaborados con *eAdventure*, mejorar el propio *eAdventure* y servir de base sólida para la construcción de nuevos proyectos *e-learning*.

uAdventure se integra en el mismo *Unity* con una estructura más compleja de como lo hacía *eAdventure* con *Java*, aunque a posteriori resulta ser una estructura más simple para el desarrollador.

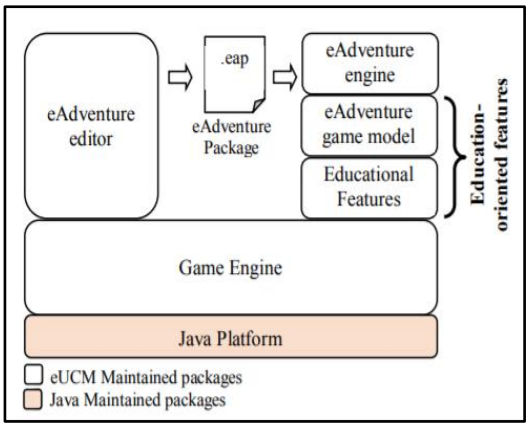

Fig. 6: Arquitectura de eA según [13] .

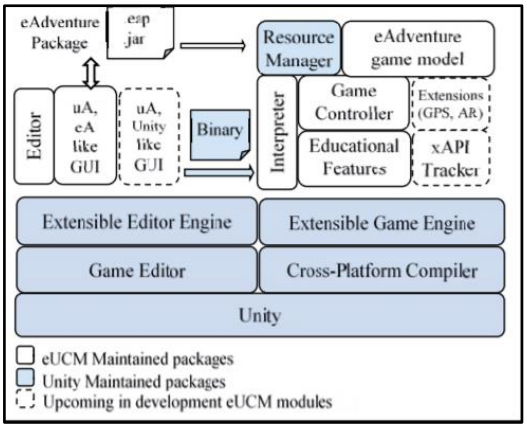

Fig. 7: Arquitectura de uA según [13].

Mientras que *eAdventure* está compuesto por múltiples capas (editor, reproductor del juego, constructor de proyectos...), *uAdventure* se compone de dos capas principales: el editor, que utilizará el docente para crear la aventura gráfica, y el intérprete, que ejecutará el juego que disfrutará el estudiante). En las figuras 6 y 7, podemos apreciar que *eAdventure* es el motor en sí y el grupo *e-UCM* se encarga de su mantenimiento, mientras que en *uAdventure* el motor es *Unity*, lo que quita carga de mantenimiento al

grupo. Al tener el editor gráfico, el intérprete y el emulador construidos sobre *Unity* (que cuenta con un editor y un motor extensibles), éstos aprovechan la arquitectura base y muchas de las capas de *eAdventure* ahora son capas estándar de *Unity*. Esto hace que *uAdventure* sea mucho sencillo y cómodo de mantener y de mejorar, ya que para modificar *eAdventure* era necesario familiarizarse con muchas más capas. En *uAdventure* se añade la posibilidad de usar *.jar* y se mantiene el formato de proyecto *.eap* para permitir proyectos de *eAdventure*.

En resumen, *uAdventure* lo forman (cajas en blanco, Fig. 7) la extensión del editor y sus interfaces (la original similar a *eAdventure* y la propuesta, similar a la propia de *Unity*), el intérprete con su controlador del juego y características educativas, y los distintos que se han añadido en los últimos tiempos o se añadirán en el futuro (como el propio módulo que ocupa este trabajo).

En cuanto a analíticas de aprendizaje, *uAdventure* permite soportar estas analíticas de aprendizaje en juegos (*GLA*) [14].

Otro de los aspectos que mejora la elección de *Unity* con respecto a *Java*, es el soporte multimedia, puesto que, en ejecución del juego, *uAdventure* utiliza los reproductores multimedia de *Unity*. Además, gracias al proyecto *FFmpeg* [15] es posible la conversión sobre la marcha del contenido multimedia. Mientras que *eAdventure* soporta algunos formatos como *MPEG1*, *DIVX* o *XVID*, entre otros, *uAdventure* soporta prácticamente todos los formatos de vídeo, tanto en el proceso de edición (*FFmpeg*) como en la reproducción del juego (*Unity*).

Cabe mencionar que en el diagrama de la estructura de *uAdventure*, con borde discontinuo, aparecen las secciones o módulos con las que se está trabajando en la actualidad o están recientemente terminadas (septiembre 2017).

Este Trabajo de Fin de Grado, tratará de añadir una nueva extensión en *uAdventure*, al igual que los módulos de Geolocalización o de códigos *QR*, con el fin de poder generar minijuegos sencillos que puedan formar parte de la aventura gráfica.

CAPÍTULO III: TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

Introducción

En este capítulo hablaremos de las distintas tecnologías y herramientas que se han utilizado durante la realización de este Trabajo de Fin de Grado, con mayor hincapié en el propio Unity que además de herramienta se puede considerar una tecnología en sí.

C Sharp (C#)

La inmensa mayoría del desarrollo del *plugin* se ha programado con este lenguaje. *C#* es una extensión de C creada a finales de los años noventa, en 1999, por Anders Hejlsberg [16]. Partiendo de *C*, surgió *C++*, cuyos signos "+" representan el incremento del lenguaje con respecto a su origen. En el caso de *C#*, pasa lo mismo, el signo "#" recuerda a cuatro signos "+" representando el incremento.

C Sharp es un lenguaje orientado a objetos, con bastantes similitudes con *Java* y fue creado con la intención de generar programas sobre la plataforma *.NET*, framework de *Microsoft*, empresa propietaria de los derechos y desarrolladora de este lenguaje [17].

Este lenguaje ha sido el utilizado para los distintos objetos y *scripts* de este proyecto, al igual que en el resto de *uAdventure* que lo usa por ser el lenguaje más capaz de los dos oficiales de *Unity 3D* (por ejemplo, C# soporta co-rutinas, mientras que *JavaScript* no).

XML (eXtensible Markup Language)

Se trata de un meta-lenguaje desarrollado por la W3C (*World Wide Web Consortium*) [18] basado en marcas que permite almacenar datos de forma legible. Se basa en el lenguaje *SGML* y permite definir la gramática de lenguajes específicos para estructurar cómodamente documentos grandes.

XML no sólo se creó para ser utilizado en Internet, su aplicación en el intercambio de información estructurada entre distintas plataformas facilita que se pueda utilizar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa.

Cada documento *XML* se puede completar con otro documento de tipo *DTD* (*Document Type Definition*) en el que se definen las distintas etiquetas del *XML*, referenciado en la cabecera. También es habitual el uso de los documentos *XML Schema* o *XSD*, que son similares al *DTD*, pero usan la misma sintaxis que el *XML*, permiten especificar tipos de datos.

En *uAdventure* se usa *XML* para almacenar los datos de configuración como, por ejemplo, los relativos al juego en creación, guardando cada objeto, escena o minijuego

añadido desde el editor. Por este motivo, para seguir una estructura y un estilo lo más cercano al resto de *uAdventure*, nuestro módulo de minijuegos utilizará *XML*.

MonoDevelop

Fig. 8: Logotipo MonoDevelop

Junto con la herramienta *Unity* suele descargarse también este editor de texto simple que aporta ciertas ventajas al trabajo con *Unity*. Este editor tiene la capacidad de abrir proyectos enteros y tener así relacionados todos los archivos de código. Se puede decir que Monodevelop [19] es un paso intermedio entre los editores simples como *Sublime Text* o *Notepad++* y los *IDE*'s como *NetBeans* o *Microsoft Visual Studio*. Pertenece a Xamarin, una filial de Microsoft, y se creó en un inicio como un *IDE* para desarrollo con *.NET*, framework de Microsoft.

Monodevelop hace sugerencias sobre posibles métodos existentes en librerías u otras clases incluidas en el proyecto. También reconoce palabras reservadas creadas por el usuario además de las propias del lenguaje de programación.

Aunque *Unity* es la herramienta que en todo momento ha estado presente durante el desarrollo, *MonoDevelop* ha sido el editor con el que se ha programado todo lo necesario para este trabajo.

Unity 3D

Fig. 9: Logotipo Unity

Esta herramienta es la base del proyecto, es decir, *uAdventure* busca reconstruir *eAdventure* sobre *Unity* y aprovechar al máximo la potencia de este motor de videojuegos disponible para *Windows*, *Linux* y *Mac*.

La primera versión fue lanzada en 2005 por *Unity Technologies* únicamente para *Mac*, aunque versión a versión ha ido mejorando hasta la última lanzada, en 2015 la 5ª versión, que es capaz de generar juegos para *Linux* e incluye diversos mecanismos y soportes que lo convierten en un motor de videojuegos bastante popular.

Junto con *Unity*, se pueden utilizar herramientas de diseño gráfico en 3D como *Blender*, *Adobe Photoshop* o *Maya*. Su motor gráfico utiliza *DirectX*, *OpenGL* e interfaces propias.

En el apartado de scripts, destacamos *MonoDevelop*, anteriormente citado, sobre el que se puede programar con *Unity Script*, *C#* (en el caso de este Trabajo de Fin de Grado) o *Boo* (lenguaje con sintaxis similar a Python). Esta herramienta viene añadida como versión personalizada desde *Unity 3.0*. Este proyecto se ha desarrollado con *Unity 5.6*.

En *Unity* encontramos múltiples librerías de assets para el diseño gráfico, en las que se incluyen texturas, colores, objetos y personajes 3D, formas geométricas y cámaras.

Un proyecto en este entorno consta de diversas carpetas y subcarpetas donde se almacena la configuración y la implementación del proyecto. En el directorio raíz hay además de algunos archivos de preferencias de usuario, diversos directorios:

- En la carpeta 'Library' se almacenan las librerías utilizadas para el proyecto.
- En la carpeta 'ProjectSettings' se incluyen archivos de configuración de las mecánicas de *Unity* (por ejemplo, 'Physics2DSettings').
- En la carpeta 'obj' encontramos archivos relacionados con el compilador.
- Por último, en la carpeta 'Assets' se almacenan los archivos de las escenas creadas y las distintas implementaciones que el desarrollador vayan añadiendo (texturas, imágenes, scripts), teniendo la opción de organizar estos añadidos como desee.

GitHub

Fig. 10: Logotipo GitHub

Esta famosa plataforma de control de versiones, fue elegida no sólo por su popularidad y desempeño, si no porque el desarrollo ya existente de *uAdventure* se encontraba controlado con *GitHub* y nos fue sencillo descargar el proyecto con el que investigar, realizar e integrar nuestro módulo *Minigames*.

GitHub es una plataforma de desarrollo colaborativo que pertenece a *GitHub, Inc.* [20] donde el código de desarrollo se almacena de manera pública (aunque también se puede

hacer privado pagando). Usa el sistema de control de versiones *Git*, y el framework *Ruby on Rails* de *GitHub, Inc.*

GitHub ofrece a sus usuarios, la opción de elegir la licencia de software libre que tiene o va a tener el software que almacene en cada repositorio, aunque si no se especifica, por defecto será privativo. También pone a disposición de cada repositorio un host gratuito en el que se ejecutará el desarrollo web de un repositorio asociado. Aunque es algo limitado, pues sólo soporta webs estáticas (no permite *PHP* o demás implementaciones que comuniquen con servidor).

Desde el primer *commit* en octubre de 2007 hasta verano de 2017, se almacenan más de 66 millones de repositorios, con más de 24 millones de usuarios y más de 117 mil empresas utilizan esta plataforma para sus proyectos.

Hoy en día es muy común en desarrolladores ver el enlace a su perfil de *GitHub* en el *Curriculum Vitae*.

CAPÍTULO IV: PROCESO DE DESARROLLO

Metodología de trabajo

A la hora de planificar el proyecto, nos hemos apoyado en los conceptos que caracterizan y fomentan las metodologías ágiles como los ciclos cortos de desarrollo y la toma de decisiones compartida y a corto plazo, pues son robustas ante posibles cambios en los requisitos.

En estos ciclos, el objetivo de cada uno era conseguir software que funcione sin errores, formando así una versión estable para incrementarla después repitiendo el proceso.

Se han realizado reuniones, cara a cara, frecuentemente entre los autores, entre 3 y 4 reuniones cortas por semana para tratar temas del proyecto, tanto planificación como avances y problemas encontrados.

También se tuvieron varias reuniones con el director Manuel Freire, semanalmente al principio, para tratar los avances en la investigación del entorno, y para ofrecer los avances con los prototipos. De igual modo se hizo con los desarrolladores de *uAdventure*, Victor Manuel e Iván.

Para introducir esta metodología en nuestro proyecto, uno de los puntos más importantes ha sido la fijación de pequeños objetivos a fin de avanzar poco a poco y de forma más segura. Esto lo aplicamos al comienzo del desarrollo empezando por cambios en la vista de editor antes de proceder a la lógica interna.

Plan de trabajo

En un principio se dedicó el primer cuatrimestre del curso a una investigación sobre el entorno *Unity* [21] ya que no contábamos con la experiencia necesaria para la realización de un *plugin* con esta herramienta. Además de la básica investigación de saber qué es *Unity*, cómo funciona o, incluso, por qué funciona, se realizaron distintas escenas sencillas a modo de toma de contacto.

Más adelante, se realizaron algunas versiones primitivas y abstractas de lo que se deseaba desarrollar: un minijuego educativo. En este periodo, se desarrolló un juego de relacionar preguntas y respuestas, que forma la base de nuestro editor de minijuegos. También se consiguió en este punto, la capacidad de guardar en un archivo *JSON* (*JavaScript Object Notation*) [22] la configuración del minijuego, es decir, las preguntas y las respuestas.

Seguidamente, desarrollamos un juego simple de diálogos entre los personajes (en este caso, piratas en una taberna). Este juego constaba únicamente de una conversación con un pirata, que reaccionaba a cada una de nuestras respuestas. Aquí aplicamos lo

aprendido con *JSON*, a pesar de que *uAdventure* utiliza *XML* para los archivos de configuración.

Posteriormente, pasamos al aprendizaje sobre la herramienta en desarrollo *uAdventure* en la cual trabajan Víctor Manuel Pérez Colado e Iván Pérez Colado. Se llevaron a cabo varias reuniones con el objetivo de aclarar diversas dudas del funcionamiento de la herramienta y de cómo se podría integrar nuestra extensión y cómo seguir el diseño establecido en el proyecto anfitrión.

Por último, se ha procedido a la codificación del *plugin*, teniendo como inspiración el resto de módulos que ya existen en *uAdventure*, como, por ejemplo, los módulos *QR Expansion*, *GeoElements* o *MapzenGo*. Para esta parte del desarrollo, se ha utilizado un sistema de etiquetas existente en *uAdventure* que ayuda a la herramienta a reconocer este módulo *Minigames* y así ejecutar el editor tanto si tenemos instalado el *plugin* como si no. Se ha seguido una estructura de archivos similar a los demás módulos, especialmente al módulo *QR Expansion*, que consiste en separar en carpetas el modelo (Model), los efectos (Effects) y las funcionalidades (*Scripts*).

Para la comprensión de este sistema, realizamos un diagrama aclaratorio (ver Fig. 11) de cómo funciona a grandes rasgos el módulo *QR Expansion* y las relaciones que tienen las distintas clases que intervienen en este módulo:

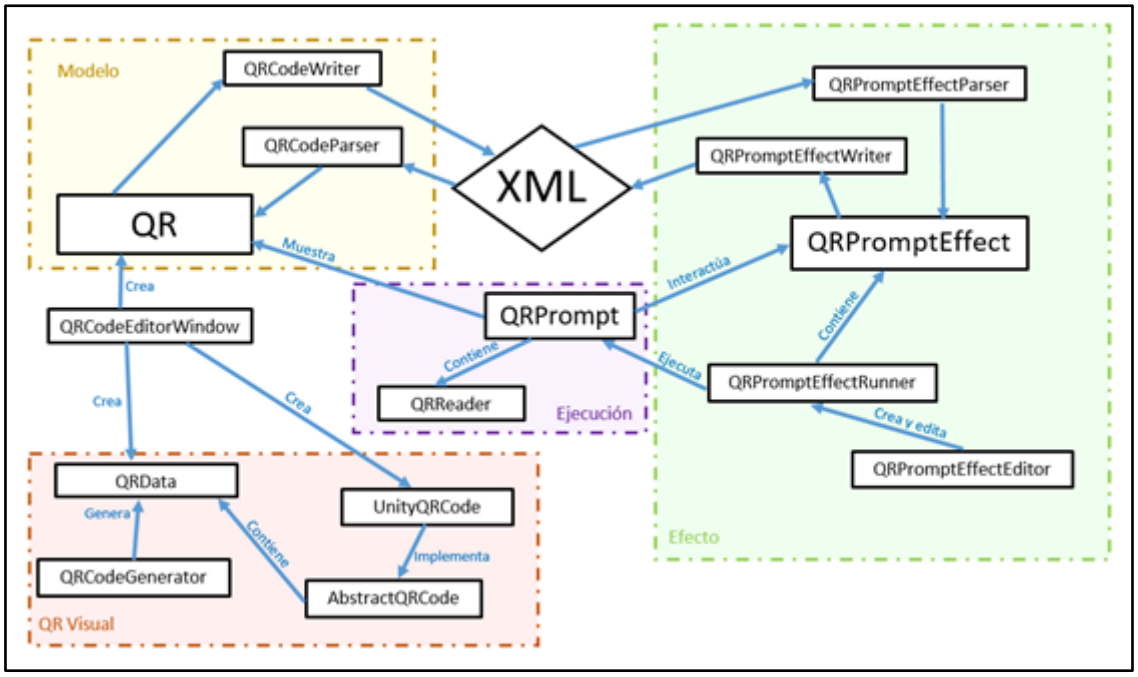

Fig. 11: Esquema de funcionamiento del módulo *QR Expansion*. Elaboración propia.

El diagrama contiene las distintas secciones que forman el módulo *QR Expansion*, que permite leer códigos QR desde la cámara conectada al ordenador.

La Tabla 1 muestra el funcionamiento de cada una de las secciones dentro del módulo *QR Expansion*, y la funcionalidad que tendrán dichas secciones en nuestro módulo de *Minigame*.

Nombre	Funcionalidad en QR Expansion	Funcionalidad en Minigame
Modelo	Guarda y carga el QR con todos sus atributos.	Guarda y carga el minijuego con todos sus atributos.
Efecto	Encargado de crear los efectos, así como guardarlos, cargarlos e iniciar su ejecución.	Encargado de crear los efectos, así como guardarlos, cargarlos e iniciar su ejecución.
Ejecución	Ejecuta el QRReader, abre la cámara para el escaneo de QR.	Ejecuta el Minijuego y carga todos los scripts necesarios para el correcto funcionamiento de éste.
QR Visual	Genera internamente el QR que ha de será leído por el QRReader.	No hay homólogo.
QrCodeEditorWindow	Ventana de editor que crea el QR y relaciona el modelo con el QR Visual.	Ventana de editor que crea el minijuego con sus preguntas y respuestas.

Tabla 1: Secciones de la implementación de QR Expansion y su reflejo en Minigame.

A partir de este diagrama, creímos oportuno realizar una división en el modelo. El objeto principal sería la clase *Minigame* que únicamente tendría un identificador, el resto de la información se encontraría almacenada en una clase llamada *MinigameData*. Esto aportaba algunas ventajas como reducir ligeramente la carga del motor al tratar sólo con un *Id*, y acceder a los datos sólo cuando fuera necesario. También benefeciaría mucho la posible ampliación de tipos de minijuego, pues los distintos *MinigameData*, heredarían de *Minigame* y tendríamos una estructura más cómoda. Esta idea se desechó finalmente debido a que el beneficio en la carga no era significativo, y que al final no se realizó la ampliación descrita. Esto, al menos, nos dio un posible punto de partida si en el futuro se quiere implementar esa mejora.

En esta parte del proceso seguimos el siguiente recorrido, procurando siempre cumplir la planificación de ciclos de desarrollo cortos.

Antes de empezar a hablar de la codificación del proyecto hay que destacar que en *uAdventure*, cada aventura gráfica es considerada un capítulo, y toda la información necesaria para cargar un capitulo (escenas, personajes, objetos, etc.) se guarda en un fichero *XML* dentro de una carpeta llamada *CurrentGame*.

Comenzamos en un primer momento por generar una extensión de la vista del editor *uAdventure* que correspondiera al módulo *Minigames*. Fijándonos en los demás módulos, estos primeros pasos nos resultaron llevaderos, aunque solo fue en la fase más básica. Lo difícil sería adaptar nuestro trabajo a la estructura de código de *uAdventure*, utilizando las mismas mecánicas y un diseño similar, en la medida de lo posible.

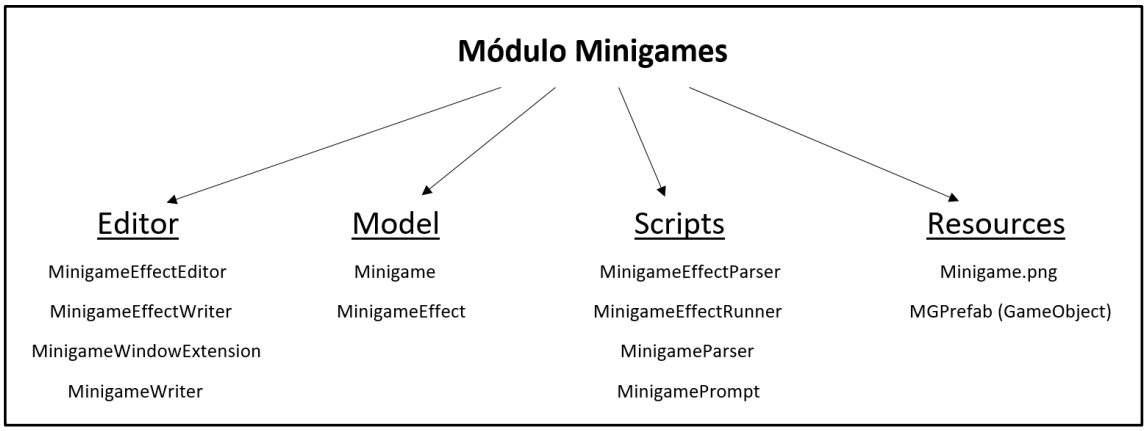

Fig. 12: Estructura de archivos del módulo Minigames.

Una vez añadida esta vista, el siguiente paso fue estructurar el módulo (ver Fig. 12) y con ello crear las clases (por el momento, sin funcionalidad) que serían necesarias para el funcionamiento del editor. Como en la mayoría del proceso, nuestra inspiración fue el *QR Expansion*, y comenzamos a desarrollar las clases que sustentan este editor: el *Writer*, para guardar la configuración en el *XML*; y el *Parser*, para leer posteriormente esa configuración.

La interacción en *uAdventure* funciona principalmente con lo que llamamos efectos, éstos realizan una acción con algún objeto de la escena (items, books, minigames), o con la escena en sí misma, como por ejemplo "*go to previous scene*". Para que los minijuegos creados tuvieran una repercusión en la aventura era necesario crear un efecto que consiga que el minijuego se ejecute, por lo que creamos una clase *MinigameEffect* con sus correspondientes *Writer* y *Parser*, que funcionan igual que en la clase Minigame: guardando la configuración en un archivo *XML* y leyéndola más adelante.

Para que el Minigame pueda ejecutarse en la aplicación es necesario crear un *MinigameEffectRunner* que mediante un sistema de etiquetas es capaz de ejecutar solo los efectos del tipo *MinigameEffect*. En el caso de *minigameEffect*, su ejecución consiste en crear y ejecutar un *MinigamePrompt*, una clase que se encarga de invocar un

GameObject previamente creado llamado *MinigamePrefab*, e introducir en él los datos necesarios para la ejecución (preguntas y respuestas).

Una vez se ejecuta el *MinigamePrompt*, el minijuego aparece en pantalla y es posible jugar gracias a los scripts internos del *GameObject* invocado.

Primero creamos un objeto *MinigameEffect*, que se tratará en el sistema de ejecución. Para este efecto era necesario implementar sus correspondientes *Writer*, *Parser* y *Runner*, de modo que procedimos a ello teniendo que investigar a fondo el tratamiento de efectos de *uAdventure*. En este caso, necesitábamos algo más que el módulo *QR*, por lo que buscamos similitudes entre lo que ya estaba implementado y nuestra extensión.

Tras aprender y entender el tratamiento de efectos en todos y cada uno de los módulos del editor, sacando denominador común conseguimos implementar nuestro tratamiento de efectos, completando correctamente las clases que actúan sobre el *MinigameEffect*, haciendo que éste implemente la clase abstracta *AbstractEffect*, y que sea de tipo *CustomEffect* (ver Fig. 13). Cada efecto en *uAdventure* hereda de la clase abstracta. Como extra se implementó una clase abstracta que serviría para las nuevas funcionalidades que se incorporen a *uAdventure*, como la geolocalización, los QR o los minijuegos, en el caso de este trabajo.

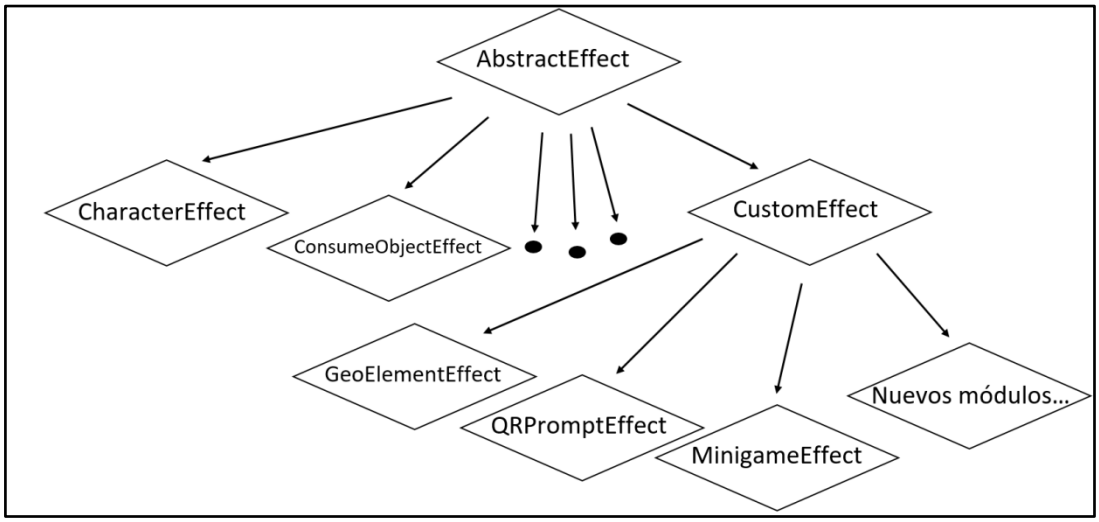

Fig. 13: Diagrama de herencia relativo a los Effects.

Junto con todo esto, implementamos la clase que serviría para la vista de crear y editar *MinigameEffects* que posteriormente se insertarán en una *Scene*: *MinigameEffectEditor*, que en nuestro caso solo consiste en seleccionar el minijuego, entre los creados previamente, que se lanzará al ejecutar ese efecto. Gracias a la herencia de MinigameEffect representada en Fig.13 conseguimos que este efecto salga como una opción cada vez que queramos insertar un efecto en cualquier parte de *uAdventure* y solamente si existe al menos un minijuego creado.

En este punto, nos encontramos con numerosos errores, la inmensa mayoría derivados de no haber profundizado tanto en el funcionamiento de todos los aspectos de la herramienta.

Internamente, *uAdventure* guarda lo relativo a cada proyecto en la carpeta *CurrentGame*, y a la hora de salvar el progreso se sobrescribe esta carpeta. En ocasiones, si se produce un error en el proceso de sobrescritura, la carpeta se elimina y el programa se detiene antes de volver a crearla. Esto provoca ciertos errores al guardar el progreso o al cargar de nuevo el proyecto, puesto que el sistema no encuentra el directorio *CurrentGame*.

Por otro lado, *uAdventure* utiliza un sistema de etiquetas, que hace uso de identificadores textuales que permiten localizar *Parsers* especializados. Este sistema recorre todos los *Parsers* de la herramienta y los ejecuta para leer el *XML* de configuración cuando se abre un proyecto, lo cual facilita mucho la labor de añadir nuevas funcionalidades. El *Writer* establece la etiqueta/identificador y el sistema busca el *Parser* con la etiqueta correspondiente. Esto nos provocó errores del tipo *NullReference* en el proceso de reconocer los *Parsers* por culpa de simples errores tipográficos (usando la etiqueta 'MinigameEffect', recogíamos 'minigameEffect').

Esta fase de errores retrasó las pruebas del sistema de efectos y, por tanto, el comienzo del desarrollo de la lógica de ejecución. Esto, aun siendo una de las partes más laboriosas (superada claramente por los *Effects*), resultó ser la fase más agradecida para la moral, pues cada avance se reflejaba visualmente de inmediato motivándonos a continuar.

Componentes y vistas

El *plugin* de este proyecto, consta de diversos scripts o clases, dependiendo de la funcionalidad u objetivo que tengan. Del mismo modo que otros módulos de *uAdventure*, el módulo *Minigames* posee un *Writer*, un *Parser*, además de la clase fundamental "Minigame.cs" y su enlace con la ejecución "MinigamePrompt.cs".

El "MinigameWindowExtension.cs" es el encargado de pintar la ventana de edición del minijuego (ver Fig. 14) y, a su vez, es el responsable de instanciar el minijuego si es nuevo. Además, se encarga de añadir la opción 'Minigames' al menú lateral de componentes, desde la cual se accederán o crearán los distintos minijuegos.

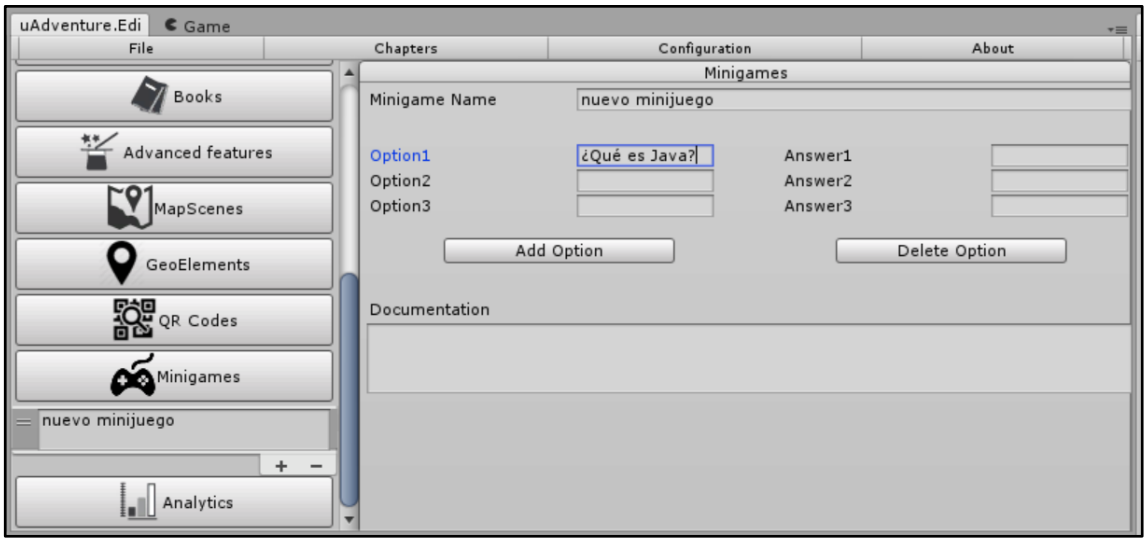

Fig. 14: Vista de editor del módulo Minigames.

El script "MinigameWriter.cs" (ver Fig. 15) nos permite guardar la documentación del minijuego en el archivo de configuración del capítulo en *XML*, desde el apartado de editor.

```
[DOMWriter(typeof(Minigame))]
public class MinigameWriter : ParametrizedDOMWriter
{
    protected override void FillNode(XmlNode node, object target, params IDOMWriterParam[] options)
    {
        var element = node as XmlElement;
        var mg = target as Minigame;
        element.SetAttribute ("id", mg.Id);
        AddNode (element, "content", mg.Content);
        AddNode (element, "documentation", mg.Documentation);
        XmlElement pre = AddNode (element, "preguntas", "");
        foreach (var i in mg.Preguntas) {
            AddNode (pre, "item", i);
        }
        XmlElement resp = AddNode (element, "respuestas", "");
        foreach (var i in mg.Respuestas) {
            AddNode (resp, "item", i);
        }
        DOMWriterUtility.DOMWrite (element, mg.Conditions);
        DOMWriterUtility.DOMWrite (element, mg.Effects);
    }
```

Fig. 15: Método que escribe sobre el XML los datos de un minijuego.

A la hora de cargar la información desde el archivo de configuración, "MinigameParser.cs" es el archivo que lo lleva a cabo (ver Fig. 16), recogiendo del archivo *XML* la información de los minijuegos ya creados para ese proyecto y poder así ejecutarlos o mostrarlos en el editor.

```
[DOMParser(typeof(Minigame))]
[DOMParser("minigame")]
public class MinigameParser : IDOMParser
{
    public object DOMParse(XmlElement element, params object[] parameters)
    {
        var mg = new Minigame(element.Attributes["id"].Value);
        mg.Content = element.SelectSingleNode("content").InnerText;
        mg.Documentation = element.SelectSingleNode("documentation").InnerText;


        XmlNode pre = element.SelectSingleNode ("preguntas");
        int cont = 0;
        foreach( XmlNode i in pre.SelectNodes ("item")){
            mg.addOption ();
            mg.setPregunta (cont, i.InnerText);
            cont++;
        }
        cont = 0;
        XmlNode resp = element.SelectSingleNode ("respuestas");
        foreach( XmlNode i in resp.SelectNodes ("item")){
            mg.setRespuesta (cont, i.InnerText);
            cont++;
        }
```

Fig. 16: Método encargado de obtener la información del archivo XML de configuración

Dentro del modelo tenemos la clase "Minigame.cs" que es la que alberga la información del minijuego (ver Fig. 17), es decir el identificador, las preguntas y las respuestas.

```
public class Minigame : HasId, Documented
{
    public Minigame(string id)
    {
        Id = id;
        Conditions = new Conditions();
        Effects = new Effects();
        Respuestas = new string[0];
        Preguntas = new string[0];
    }
```

Fig. 17: Constructor del objeto Minigame.

Por otro lado, como ya se explicó en el apartado "Plan de trabajo", *uAdventure* trabaja con *Effects* para lanzar y ejecutar las distintas acciones. En nuestra extensión son necesarios estos efectos y además de incluir en el modelo el "MinigameEffect.cs", hemos generado los archivos "MinigameEffectParser.cs", "MinigameEffectWriter.cs" y "MinigameEffectRunner.cs", este último encargado de lanzar el *Effect*, con la vista "MinigameEffectEditor.cs" (ver Fig. 18).

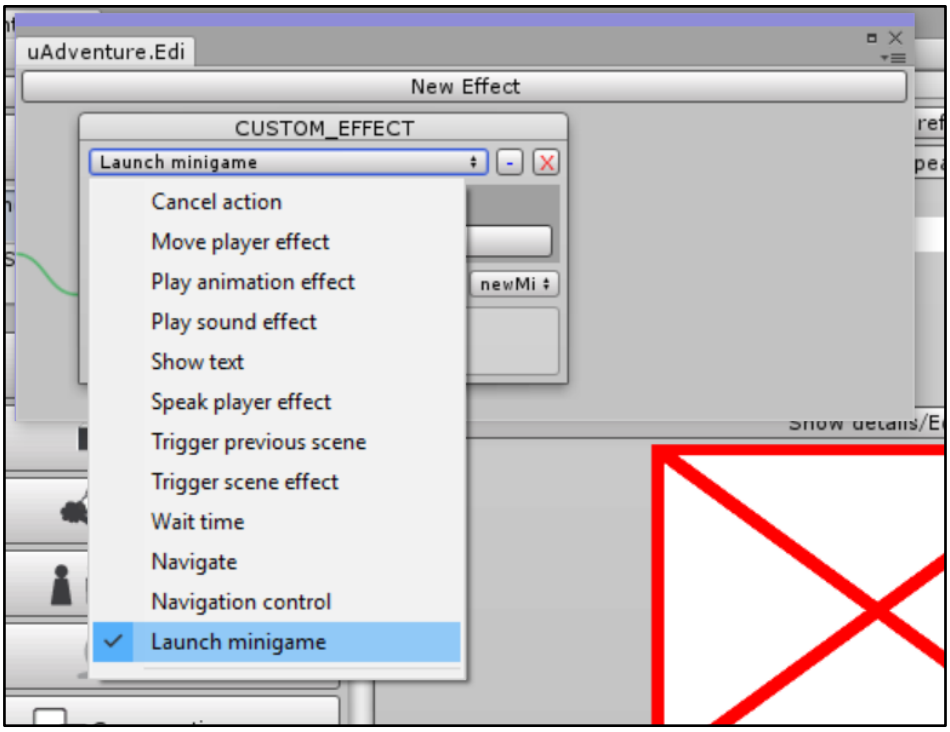

Fig. 18: Vista relativa a la inserción de un efecto en una escena.

Para la ejecución, hemos añadido un "MinigamePrompt.cs" que crea e inicializa todos los *GameObjects* necesarios para la ejecución del minijuego (ver Fig. 19). Esto es, genera la vista que se apreciará al ejecutar el minijuego dentro del juego de *uAdventure*.

```
void Start()
{
    bool[] sols = new bool[mg.Respuestas.Length];
    for (int a = 0; a < sols.Length; a++) {
        sols [a] = false;
    }
    preg=new GameObject[mg.Preguntas.Length];
    resp=new GameObject[mg.Respuestas.Length];
    //spawn object
    for(int i=0;i<mg.Preguntas.Length;i++){
        preg [i] = new GameObject("Pregunta"+i.ToString());
        preg [i].AddComponent<TextMesh> ();
        preg [i].GetComponent<TextMesh> ().text = mg.Preguntas [i];
        preg [i].AddComponent<BoxCollider2D>();
        preg [i].GetComponent<BoxCollider2D> ().isTrigger = true;
        preg [i].transform.position = new Vector3 (-13,(-5*i)+9, 0);

    }


    for (int i = 0; i < mg.Respuestas.Length; i++) {
        bool respondida = true;
```

Fig. 19: Extracto de código de la clase "MinigamePrompt".

Resultados

En este apartado vamos a exponer el resultado de este proceso de desarrollo. Hablaremos de las capacidades que nos ha otorgado toda la investigación sobre *uAdventure*, de la composición final del *plugin* de minijuegos y de su instalación y uso.

Las innumerables pruebas que hemos realizado sobre *uAdventure*, nos han ayudado a comprender casi por completo el funcionamiento interno de la herramienta. Ahora conocemos cómo trabaja el sistema de efectos y el de etiquetas. Estos conocimientos son básicos si se quiere desarrollar un módulo nuevo en *uAdventure* y por lo tanto el apartado de integración en la herramienta estaría superado.

Tanto el módulo *QR Expansion* como ahora *Minigames*, contienen en su directorio todo lo necesario para funcionar. Así el comportamiento único de estas extensiones, depende únicamente de los archivos e imágenes que aporte el propio módulo. Con esta estructura de *uAdventure* se facilita mucho la inserción de nuevas funcionalidades o secciones para el juego.

Para la instalación de este módulo es necesario la incorporación del directorio Minigames en la carpeta '*Assets*' de *uAdventure* (ver Fig. 20), y abrir este proyecto con *Unity 3D*.

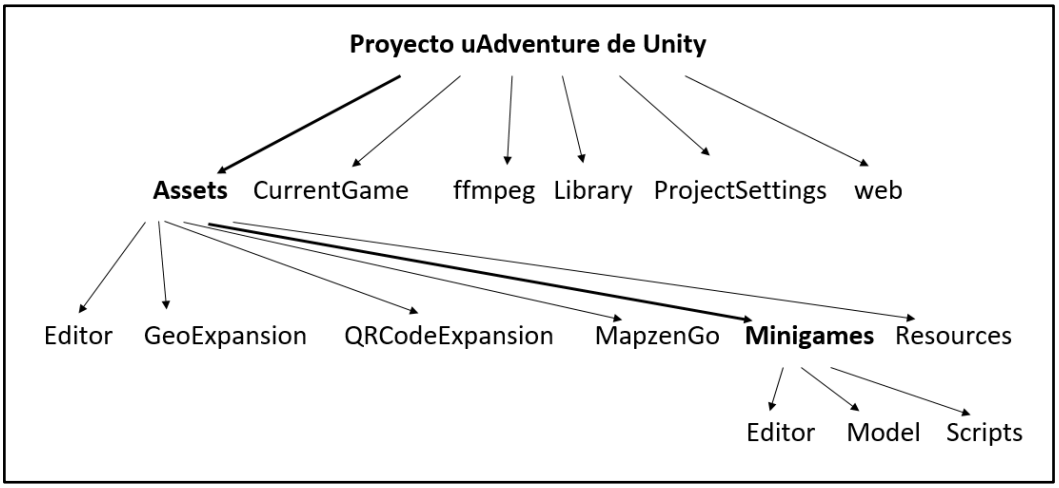

Fig. 20: Estructura de directorios del proyecto uAdventure y el módulo Minigames.

En el editor de *uAdventure*, se ha añadido la opción de crear un minijuego al menú desplegable de componentes. Al utilizarla abrimos la sección *Minigames* de la barra lateral, y añadimos un nuevo minijuego pulsando en el icono '+' (podremos eliminarlo con el icono '-') (ver Fig. 21). Tras seleccionarlo podemos observar que se abre el editor propio de este módulo.

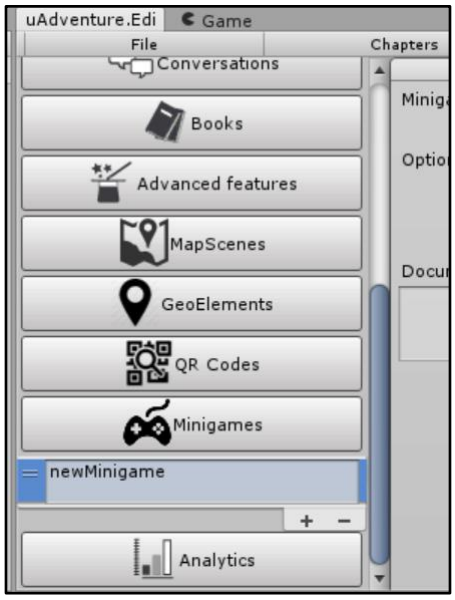

Fig. 21: Sección del editor relativa a Minigames.

En él, nos encontramos un campo para definir el nombre del minijuego (por defecto 'newMinigame'), dos botones (Añadir y eliminar opción) y el campo de texto de 'Documentation' en el que podemos añadir una descripción del minijuego. Con el uso de los botones podemos hacer aparecer campos de texto destinados a introducir preguntas y respuestas.

Una vez creado el minijuego deseado, para incluirlo en una escena del juego, debemos insertar un MinigameEffect. Para ello, nos vamos a la sección de escenas, y en la pestaña *Exits* añadimos el efecto con el texto 'Launch Minigame' y elegimos el minijuego que queremos ejecutar en esa escena (ver Fig. 22).

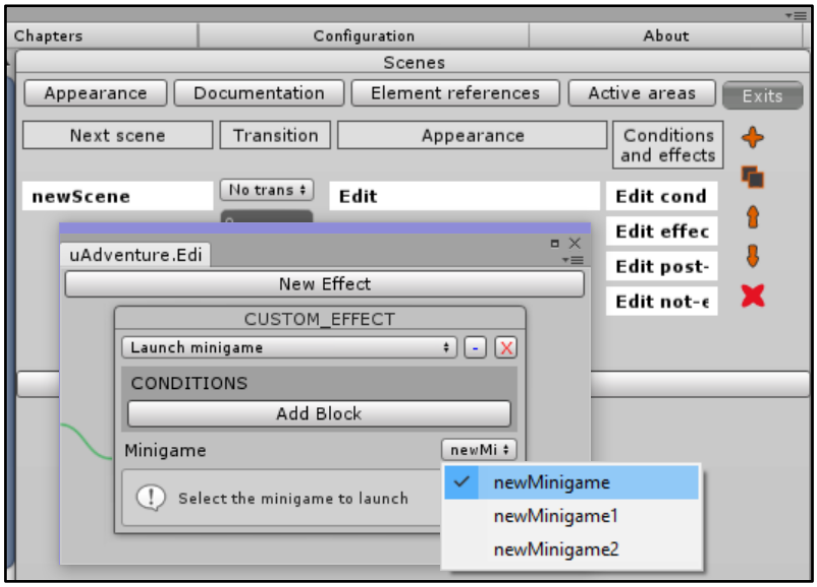

Fig. 22: Inserción de un MinigameEffect en una escena.

Una vez añadido, al ejecutar el juego y llegados a esa escena, clickando en el área indicada, se abrirá nuestro minijuego mostrando las preguntas y respuestas que hemos añadido (ver Fig. 23). Para jugarlo, sólo es necesario arrastrar las respuestas a su correspondiente pregunta.

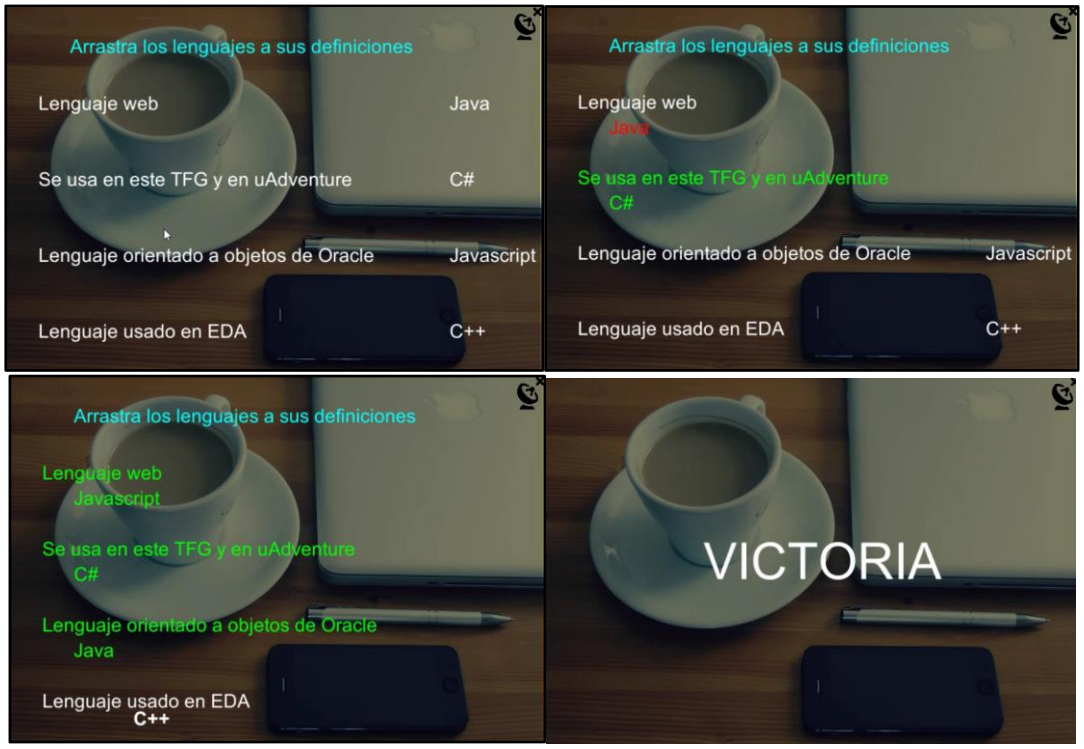

Fig. 23: Secuencia de ejecución del minijuego creado.

Gracias a este largo proceso de desarrollo, hemos aprendido a manejar con cierta soltura la herramienta *uAdventure*, y al mismo tiempo a crear un *plugin* para esta herramienta, lo cual podemos aplicar a proyectos similares. Ahora tenemos conocimiento de las pautas a seguir para desarrollar una extensión de este tipo, así como de lo necesario para evitar un gran número de errores que pueden retrasar los plazos de entrega.

Otro aspecto importante que hemos ganado es la experiencia de haber trabajado con software ajeno, y de gran envergadura (97 *commits* en *GitHub*), algo inédito durante los años de carrera anteriores. Esto puede ser muy valioso de cara a nuestra vida laboral.

Una de las cosas aprendidas, que puede servir en cualquier entorno de desarrollo, es la importancia de familiarizarse en profundidad con el motor de *Unity 3D*, así como con la estructura de sus proyectos. También es necesario estudiar a fondo el funcionamiento de plataformas como *GitHub*, dado que es muy sencillo corromper un proyecto si se incluyen archivos auto-generados por *Unity* o cualquier otra herramienta de desarrollo.

En el archivo ‘readme.md’ del repositorio, se detallan las instrucciones de instalación y utilización del *plugin*, así como el proceso a seguir en caso de querer añadir

funcionalidad al módulo *Minigames*, y las instrucciones para crear nuevos módulos para *uAdventure*.

El resultado de todo este proceso, cumpliendo así el objetivo principal de este trabajo (Ob.1), además de un *plugin* compatible con *uAdventure* para crear y editar minijuegos de relacionar preguntas y respuestas, es una guía documentada con diversos consejos útiles para el desarrollo de nuevos módulos para *uAdventure* que podrán utilizar futuros programadores.

El código de *uAdventure* con el módulo instalado se encuentra alojado en:

https://github.com/uAdventureMinigames/uAdventure

CAPÍTULO V: CONCLUSIONES

En este capítulo, expondremos ideas que se desecharon, implementaciones o procesos que se pudieron realizar de otro modo, posibles mejoras que se podrían implementar en el futuro, el aprendizaje obtenido y, por último, las conclusiones que hemos sacado tras la realización de este trabajo.

Post mortem

Durante todo el proceso, hemos ido solucionando problemas que, a posteriori, resultan mejor abordables de otro modo. Estos modos, que en su día no tuvimos en cuenta o desechamos, hemos querido mencionarlos, pues probablemente nos hubieran ahorrado tiempo de desarrollo o mejorado el resultado final de nuestro *plugin*.

En repetidas ocasiones, Víctor Manuel e Iván nos ofrecieron su ayuda en cuanto al desarrollo software incluso con la opción de programar juntos ciertas partes. En el caso de haber aprovechado mejor esta oferta, algunos de los errores o problemas que nos han surgido, es posible que ellos ya hubieran tratado con casos similares, y habríamos ahorrado tiempo y mejorado el código.

Esta forma de trabajar, no solo podría habernos ayudado a rebajar tiempo de proceso si no que hubiera servido de *testing* para el desarrollo de *uAdventure*, contribuyendo de este modo a la mejora del proyecto, más allá de nuestra competencia.

Otra de las opciones de la planificación que no tuvimos en cuenta fue en la fase de investigación el entorno. Al mismo tiempo que profundizamos en el funcionamiento de *Unity*, podríamos haber comenzado la toma de contacto con *uAdventure* y así haber avanzado en la comprensión del proyecto, lo que puede que nos hubiera ahorrado tiempo y evitado estancamientos innecesarios. En varias ocasiones, estos estancamientos nos han obligado a recuperar una versión de *uAdventure* limpia de cualquier rastro de nuestro trabajo, puesto el simple acto de deshacer cambios no era suficiente para arreglar el problema en cuestión. De este modo, era necesario empezar de nuevo a incluir poco a poco nuestras aportaciones. A golpe de reiniciar y de excepciones, finalmente hemos aprendido con bastante soltura el funcionamiento interno de *uAdventure*.

Por último, es interesante mencionar la idea desechada de un hipotético *MinigameData*, explicado anteriormente (ver ‘Plan de trabajo’, pág. 23).

Trabajo futuro

Este *plugin* se puede mejorar de diversas formas.

Nuestro código, así como en general el código de *uAdventure*, es ampliable, cumpliendo el objetivo marcado para facilitar el trabajo a futuros desarrolladores (Ob.2). Esto quiere decir que se compartimenta bien la lógica de forma que en un futuro se pueda añadir nuevos módulos, o nuevos tipos de minijuegos, en el caso de nuestro módulo *Minigames*. Esto sería posible haciendo uso de la herencia. Cada tipo de minijuego tendrá su clase modelo (implementando la clase abstracta Minigame) con su *Writer* y su *Parser*, su propio *MinigamePrompt* para la ejecución, y, por supuesto, su propia ventana de editor, extendiendo la ya existente del mismo modo que ésta extiende la de *uAdventure*. Toda la lógica de efectos sería común y gran parte del código nuevo reutilizaría código ya implementado.

Otra posible mejora sería añadir una opción para deshacer los cambios y volver a la versión del minijuego anterior, o a la versión vacía, y otra para guardar los nuevos cambios definitivos (ver Fig. 24). En la versión actual hemos querido mantener la estética global de *uAdventure* que guarda la información de los distintos objetos desde cada campo de texto y luego la configuración completa en el *XML* con la opción "*Save project*" del menú "*File*" del editor.

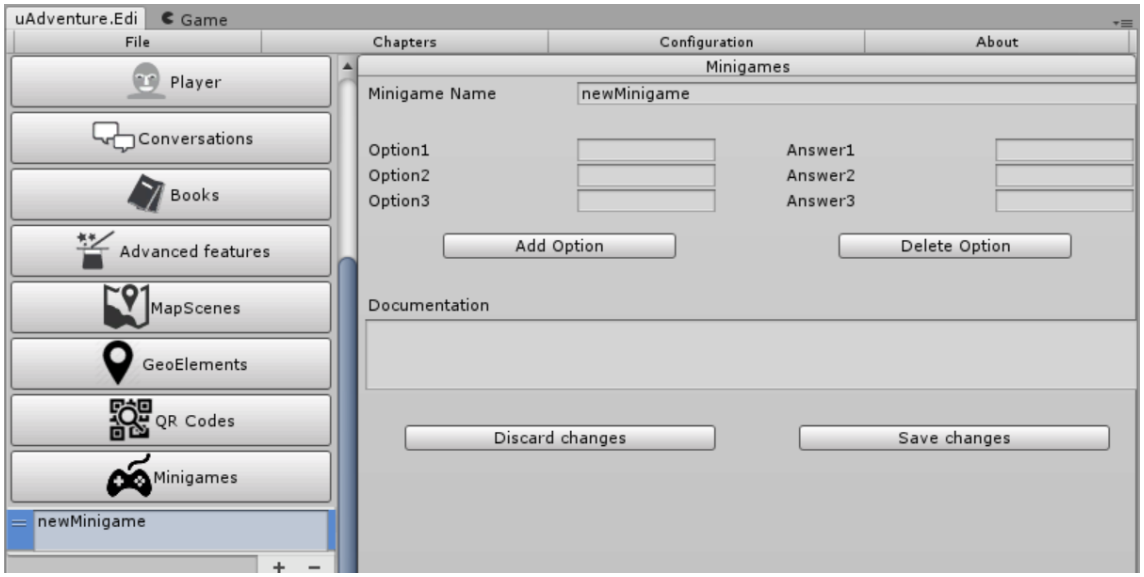

Fig. 24: Vista de editor de minijuegos con opciones de guardar o descartar cambios.

Por último, pensamos que puede ser útil una buena batería de pruebas a fin de testear minuciosamente el resultado final para darle más estabilidad a este *plugin* y que se integre de la mejor manera posible dentro del resto de la herramienta *uAdventure*.

Competencias adquiridas

Este proyecto nos ha ayudado (y en ocasiones obligado) a aprender a programar eficientemente y bien estructurado, en un lenguaje como *C#* en conjunto con la herramienta de *Unity*, ya que programar para modificar el comportamiento de una herramienta fue completamente nuevo para nosotros.

También nos ha enseñado a compenetrarnos como equipo y a comunicarnos con otros desarrolladores para encaminar correctamente nuestro trabajo y así poder incluirlo cómodamente en el proyecto anfitrión. De este modo cumplimos también el objetivo inicial de mejorar nuestras prácticas y nuestro entendimiento como programadores (Ob.3).

Nuestra percepción del código ajeno ha mejorado considerablemente. Después de leer, ejecutar y comprender el código ya desarrollado (135.539 líneas de código en 97 *commits* sin el *plugin Minigame* en septiembre de 2017) hemos aprendido nuevos estilos de programación gracias también a la comunicación con el equipo de *uAdventure* y al uso de plataformas como *GitHub*.

Como ocurre con cualquier interacción con software ajeno, esto nos ha enriquecido notablemente para nuestro futuro como programadores.

Conclusión

Durante el desarrollo de este *plugin* nos hemos dado cuenta de lo difícil que resulta modificar o ampliar un código ajeno. Más difícil todavía si este código es un proyecto tan grande como lo es *uAdventure*, que además se integra con el funcionamiento de una herramienta bastante compleja como lo es *Unity*.

El primer problema con el que nos encontramos fue descubrir la dificultad de este proyecto, no solo por su complejidad intrínseca, sino por la inexperiencia que teníamos en el campo de desarrollo usando *Unity*. Esta ignorancia nos condujo a pasar gran parte del primer cuatrimestre del curso aprendiendo a usar *Unity* de la forma más básica, y entender, a nivel usuario, cómo funcionarían las cosas durante el desarrollo.

Este proyecto nos resultó un reto excelente y atractivo, que podría ayudarnos a formarnos como programadores y a sobrellevar la carga que supone un proyecto en conjunto con otro equipo de desarrollo. Con esto hemos descubierto la gran importancia que tiene la comunicación entre equipos.

Pasado el tiempo, tras descubrir las dificultades de trabajar con código ajeno, nos suponía un logro considerable cada uno de los avances. Esto indica, claramente, la importancia de fijar metas a corto plazo y pactar objetivos pequeños y reuniones frecuentes, es decir, las ventajas de una metodología de proceso ágil.

En resumen, a pesar de las dificultades encontradas, hemos conseguido de forma satisfactoria contribuir, mediante la inclusión de nuestro módulo, en un proyecto serio y de gran calibre como es *uAdventure*.

Conclusions

During the development of this plugin, we have realized how difficult it is to modify or extend foreign code. It is even harder coming from a project as big as *uAdventure*, which in turn is integrated in a complex tool like *Unity*.

The first problem we encountered was discovering the scope of this project, not only because of its intrinsic complexity, but also due to the inexperience we had in *Unity* development. This ignorance led us to spending much of the first semester of the year learning to use *Unity* in the most basic way and understanding, at user level, how things would work during development.

This project was an excellent and attractive challenge, which could help us grow as programmers and cope with the burden of a project in collaboration with another development team. Thus, we have discovered the importance of communication between teams.

After some time, each stride meant a considerable achievement. This clearly indicates the importance of setting short-term goals and agreeing on small goals and frequent meetings, i.e. the advantages of an Agile Methodology.

In summary, despite the problems encountered, we have succeeded in contributing, through the inclusion of our module, to a serious and large-scale project like *uAdventure*.

Contribución de los participantes

En este apartado, se expondrá resumidamente la contribución de cada participante dentro de este Trabajo de Fin de Grado.

Javier Sandoval Ferrandis

Descarga de la herramienta *Unity 3D* para familiarizarse con el entorno.

Realización de varias escenas simple con los objetos predefinidos de *Unity 3D* (*Ethan*, coche, árboles etc.) y cambiando parámetros de éstos, cambiar la ejecución de la escena levemente, como, por ejemplo, consiguiendo que *Ethan* saltase más o fuese más rápido.

Investigación de las diferencias de funcionamiento interno entre proyectos 2D y proyectos 3D en *Unity*.

Prototipo de un minijuego basado en preguntas y respuestas construido a base de *GameObject* desde el editor de *Unity 3D*. Más adelante desarrollamos este mismo minijuego generado a partir de un único script. En él, hice especial hincapié en mejorar la lógica para conseguir que el minijuego fuese lo más eficiente posible. Además, conseguí que las preguntas y respuestas se cargase de un fichero externo.

Creación de otro minijuego sencillo que trata de entablar conversaciones entre personajes, y varía según la opción que se elija. Este minijuego también era posible cargarlo desde un archivo interno mediante el uso de *JSON*.

Fase de reuniones con Víctor Manuel e Iván en las que se trató:

a) Origen de *uAdventure*.
b) Funcionamiento de la herramienta
c) Sistema de etiquetas, de efectos y utilidad de *Writers*, *Parsers*, etc...

Planificación del desarrollo software con plazos breves y objetivos pequeños.

Pruebas con *uAdventure* de creación de aventuras gráficas muy sencillas a fin de comprender el funcionamiento de la herramienta.

Esquematización del módulo *QR Expansion* a fin de aplicar una estructura similar en nuestro módulo *Minigames*.

Desarrollo de las clases *MinigameEffect*, y sus correspondientes *MinigameEffectParser* y *MinigameEffectWriter* para almacenar la información en el fichero *XML*. Además, creé las clases *MinigameEffectRunner* y *MinigameEffectEditor*, que se encargan de ejecutar el efecto y de la creación de éste, respectivamente

Desarrollo del apartado de ejecución con la clase *MinigamePrompt*.

Mejoras de la clase *MinigamePrompt* a fin de mejorar el aspecto visual en la ejecución.

Recapitulación de ideas desechadas y redacción del apartado “Post mortem”, así como el resumen y la conclusión.

Recogida de la información relativa a las diferentes fases del proceso para la redacción del capítulo de cuaderno de bitácora.

Redacción de la descripción detallada de cada una de las herramientas y tecnologías que hemos utilizado.

Recapitulación de ideas desechadas y redacción del apartado “Post mortem”, así como el resumen y la conclusión.

David Martín-Maldonado Jiménez

Descarga de la herramienta *Unity 3D* para familiarizarse con el entorno.

Realización de una escena 3D en la que *Ethan*, personaje predefinido de *Unity 3D*, se mueve con libertad y existe la capacidad de encender y apagar las farolas del entorno con una tecla. Aprendizaje de los *assets* predefinidos de Unity para el control de personajes, y tratamiento de texturas y formas.

Investigación de las diferencias de funcionamiento interno entre proyectos 2D y proyectos 3D en *Unity*.

Desarrollo de una escena en 2D, en la que *Super Mario* puede recorrer un mapa recogiendo monedas. Aprendizaje del funcionamiento de los *colliders* y diferencia entre proyectos 2D y proyectos 3D.

Prototipo básico del minijuego de relacionar preguntas y respuestas, creando *GameObjects* desde el editor de *Unity 3D*, sacando provecho al manual de métodos de *Unity* en el que encontré los diferentes observadores que utilizan las colisiones en ejecución. Descubrimiento del método *ScrenToWorldPoint* con el que se puede obtener la posición del puntero para que la respuesta seleccionada se mueva con el ratón.

Mejora del prototipo añadiendo sistema de colores para el final del minijuego y mensaje final de 'Victoria'. Aprendizaje del acceso desde *scripts* a las propiedades de los *GameObject*s.

Fase de reuniones con Víctor Manuel e Iván en las que se trató:

a) Origen de *uAdventure*.
b) Funcionamiento de la herramienta
c) Sistema de etiquetas, de efectos y utilidad de *Writers*, *Parsers*, etc...

Planificación del desarrollo software con plazos breves y objetivos pequeños.

Realización de la vista de editor no funcional del módulo *Minigames* en la clase *MinigameWindowExtension*.

Creación del modelo *Minigame* e introducción de los métodos para añadir o eliminar tuplas pregunta-respuesta, redimensionando las listas. Funcionalidad en la vista de edición.

Desarrollo de las clases *MinigameWriter* y *MinigameParser* para almacenar y recuperar la información del *XML*.

Desarrollo del apartado de ejecución con la clase *MinigamePrompt*.

Investigación de todo lo concerniente al grupo *e-UCM* y los *Serious Games* para la redacción de la memoria. Recogida de referencias a éstos y al proyecto *eAdventure* que posee su propia web.

Lectura y comprensión de varios artículos y tesis relacionadas con *uAdventure*, proporcionados por Manuel, y también sobre la solución de analíticas de juegos serios del grupo *e-UCM*, como los proyectos *RAGE* o *xAPI* [23].

Redacción del contexto del trabajo en la memoria con ayuda de los artículos anteriores y las numerosas explicaciones de Manuel, Víctor e Iván, durante las reuniones.

Recapitulación de ideas desechadas y redacción del apartado "Post mortem", así como el resumen y la conclusión.

Redacción de esta memoria, revisión y corrección de errores ortográficos y conceptuales.

BIBLIOGRAFÍA

[1] Definición de Minijuego [en línea]. GamerDic, Diccionario online de términos sobre videojuegos y cultura gamer, 2013. http://www.gamerdic.es/termino/minijuego Consultado en septiembre del 2017.

[2] Abt, Clark: '*Serious Games*', New York: Viking Press, 1970.

[3] Jung, Robert: '*The Army Battlezone Q & A*'. http://archive.li/yDKgF Consultado en septiembre de 2017.

[4] Julián Álvarez, Olivier Rampnoux: "*Serious Game: Just a question of posture?*", en Artificial & Ambient Intelligence (AISB '07), 2007, pp. 420-423. Disponible en: https://www.researchgate.net/publication/260517048_Serious_Game_just_a_question_of_posture

[5] "*Grupo e-UCM*" http://www.e-ucm.es/ . Última visita: septiembre de 2017.

[6] B. Fernández Manjón: "Juegos serios y analíticas de aprendizaje". Grupo e-UCM. Universidad Complutense de Madrid. Zaragoza 17/11/2015. Disponible en: https://es.slideshare.net/BaltasarFernandezManjon/juegos-serios-y-analticas-de-aprendizaje-55229236

[7] A. Serrano Laguna: "*Mejorando la evaluación de juegos serios mediante el uso de analíticas de aprendizaje*". Universidad Complutense de Madrid, Madrid, 2017.

[8] Grupo e-UCM: "Proyecto eAdventure". http://e-adventure.e-ucm.es/ . Última visita: septiembre de 2017.

[9] Peter Bright (15 de abril de 2015). "*Chrome starts pushing Java off the Web by disabling plugins*". *Ars Technica*. Última visita: agosto 2017.

[10] Peter Bright (9 de octubre de 2015). "*Firefox dropping NPAPI plugins by the end of 2016—except for Flash*". *Ars Technica*. Última visita: agosto 2017.

[11] Peter Bright (27 de enero de 2016). "*Oracle deprecates the Java browser plugin, prepares for its demise*". *Ars Technica*. Última visita: agosto 2017.

[12] "*Unity 3D*". Unity Technologies. https://unity3d.com/ .

[13] I. Pérez Colado, V. M. Pérez Colado, I. Martínez Ortiz, M. Freire, B. Fernández Manjón (2017): "*uAdventure: The eAdventure reboot - Combining the experience of commercial gaming tools and tailored educational tools*". IEEE Global Engineering Education Conference (EDUCON), 25-28 April 2017, Athens, Greece. http://hdl.handle.net/1820/7618 .

[14] I. J. Pérez Colado, V. M. Pérez Colado, M. Freire Morán, I. Martínez Ortiz, B. Fernández Manjón: "*Integrating learning analytics into a game authoring tool*". Universidad Complutense de Madrid, Madrid, 2017.

[15] F. Bellard, M. Niedermayer: "*FFmpeg Project*". . 2000 https://www.ffmpeg.org . Última visita: septiembre 2017

[16] "Anders Hejlsberg". Wikipedia. https://en.wikipedia.org/wiki/Anders_Hejlsberg . Consultado en septiembre de 2017.

[17] "*C Sharp*". Microsoft. https://docs.microsoft.com/es-es/dotnet/csharp/csharp .

Consultado en septiembre de 2017.

[18] "*Extensible Markup Language*". W3C. https://www.w3.org/XML/ . Consultado en septiembre de 2017.

[19] "*Monodevelop*". http://www.monodevelop.com/ . Consultado en septiembre de 2017.

[20] "*GitHub*". GitHub, Inc. https://github.com/ . Última visita en septiembre de 2017.

[21] "*Unity 3D Manual*". Unity Technologies. Última visita, mayo de 2017. https://docs.unity3d.com/es/current/Manual/ .

[22] "*JavaScript Object Notation*". Douglas Crockford. http://www.json.org . Consultado en septiembre de 2017.

[23] Artículos y tesis del proyecto RAGE. http://dspace.ou.nl/handle/1820/6019 . Última visita en septiembre de 2017.